PAPER · OPEN ACCESS

Rapid likelihood free inference of compact binary coalescences using accelerated hardware

IOP Publishing Mach. Learn.: Sci. Technol. 5 (2024) 045030 https://doi.org/10.1088/2632-2153/ad8982

MACHINE LEARNING
Science and Technology

OPEN ACCESS

RECEIVED
26 July 2024

REVISED
8 October 2024

ACCEPTED FOR PUBLICATION
21 October 2024

PUBLISHED
30 October 2024

PAPER

Rapid likelihood free inference of compact binary coalescences using accelerated hardware

D Chatterjee[1,2,*], E Marx[1,2], W Benoit[3], R Kumar[4], M Desai[1,2], E Govorkova[1,2], A Gunny[1,2], E Moreno[1,2], R Omer[3], R Raikman[1,2], M Saleem[3], S Aggarwal[3], M W Coughlin[3], P Harris[1] and E Katsavounidis[1,2]

[1] Department of Physics, MIT, Cambridge, MA 02139, United States of America
[2] LIGO Laboratory, 185 Albany St, MIT, Cambridge, MA 02139, United States of America
[3] School of Physics and Astronomy, U. Minnesota, Minneapolis, MN 55455, United States of America
[4] Department of Aerospace Engineering, IIT Bombay, Powai, Mumbai 400076, India
[*] Author to whom any correspondence should be addressed.

E-mail: deep1018@mit.edu

Keywords: Likelihood-free Inference, Self-supervised learning, Gravitational waves, Multi-messenger astronomy

Abstract

We report a gravitational-wave parameter estimation algorithm, `AMPLFI`, based on likelihood-free inference using normalizing flows. The focus of `AMPLFI` is to perform real-time parameter estimation for candidates detected by machine-learning based compact binary coalescence search, `Aframe`. We present details of our algorithm and optimizations done related to data-loading and pre-processing on accelerated hardware. We train our model using binary black-hole (BBH) simulations on real LIGO-Virgo detector noise. Our model has $\sim$6 million trainable parameters with training times $\lesssim$24 h. Based on online deployment on a mock data stream of LIGO-Virgo data, `Aframe` + `AMPLFI` is able to pick up BBH candidates and infer parameters for real-time alerts from data acquisition with a net latency of $\sim$6 s.

1. Introduction

It has been almost a decade since the discovery of gravitational waves (GWs) from compact binary mergers [1], with the last few years seeing a steady increase in the number of discovered GW events. While the first observing run of the Laser Interferometer Gravitational-wave Observatory (LIGO) reported only three events [1, 2][5], the number count stood at 90 within a span of five years [3]. Furthermore, the current ongoing fourth observing run (O4) of ground-based observatories LIGO/Virgo/KAGRA (LVK) has already reported more than one hundred events discovered online[6]. The trend is expected to continue with the instrument getting closer to design sensitivity in fifth observing run[7].

In parallel, the scope of multi-messenger astronomy (MMA) with GWs has seen a steady increase in terms of effort and infrastructure being invested for the joint follow-up of GW signals with electromagnetic (EM) and other high-energy astrophysical counterparts. The online alert infrastructure of the LVK currently reports GW discoveries along with follow-up data products in $\sim$30 s after merger time [4]. Early-warning searches [5] that can potentially pick up low-mass BNS systems up to $\sim$1 min before merger have been deployed online [6]. The alert distribution mechanisms, like NASA GCN[8], have seen upgrades [7], and new alert brokers like SCiMMA[9] have become available for the community to use. Publicly available services like TreasureMap [8] have seen a steady adoption from observatories to share observed and scheduled fields to

[5] GW151012, initially labeled as low-significance, was later confirmed as a third event in O1.
[6] See https://gracedb.ligo.org/superevents/public/O4/ for the most updated list.
[7] See https://observing.docs.ligo.org/plan/ for observing plans.
[8] https://gcn.nasa.gov/.
[9] https://scimma.org/.

orchestrate observations. Tools like SkyPortal [9] and TOM-Toolkit [10] have been developed to aid target-of-opportunity followup.

All this development comes at a time when the number of GW discoveries have significantly increased corresponding to the improvement in sensitivity of Advanced LIGO [11, 12], Advanced Virgo [13], and KAGRA [14] instruments, and the sensitivity of time-domain telescope facilities allow for unprecedented discovery rates. However, identifying GW counterparts jointly have been extremely challenging. The discovery of GWs and multi-wavelength EM emission from the merger of the binary neutron star (BNS), GW170817, [15, 16] remains the first and only success story, albeit a rarity, with most subsequent candidates likely to be at much farther distances [17].

One primary step toward improving follow-up campaigns is the availability of fast, real-time Bayesian parameter estimation (PE) of compact binary coalescence (CBCs) to provide accurate data products for GW follow-up. The computationally expensive part of stochastic sampling techniques, like nested sampling currently in use, involve the repeated computation of the likelihood. Techniques like reduced-order-quadrature (ROQ) [18, 19] and focused-ROQ [20] have been developed in view of making real-time PE as fast as possible. This is currently used in LVK to deliver update alerts from Bayesian parameter estimation on the timescale of several minutes to hours. Other techniques like the use of accelerated hardware for stochastic sampling has been reported in [21, 22], and mesh-free approximation for sky-localization, reported in [23, 24].

More recently, likelihood-free inference (LFI) using variational methods, have emerged as a different paradigm with flexible neural network approximators being used to learn the posterior or the likelihood. Their use has been demonstrated on GW data [25, 26], in particular with posterior estimation using normalizing flows [27], such as in the DINGO algorithm. However, in order to relay discovery alerts for prompt followup, the combination of *search and inference* needs to be considered together[10]. Also considering a live, real-time system design, several overhead costs like data transfer, file input/output operations, communicating data to a remote server, and so on are often overlooked in isolated analyses, but show up in the overall time-to-alert. It is also worth highlighting that traditionally in GW data analysis, the search and PE components have been treated separately—match-filtering searches pick up the candidates from the data stream using suitable detection statistic, and also provide important context like the best matching template and the signal-to-noise ratio (SNR) time series, which is then used to compute sky-localization maps [29] and EM-bright source properties [30] that are sent out in the sub-minute alerts by the LVK [4]. The results are then updated based on Bayesian PE results, in few hours timescale.

Although machine-learning techniques like LFI bring promise, large model size and/or long training times can be a barrier for operations. Also, given the slowly changing background over the course of days to weeks, the algorithm should be re-trainable from a previous model state, preferably without investing on expensive and dedicated online hardware for this purpose. This is currently lacking for online models like DINGO, which report 10-day training time on a NVIDIA A100 GPU [27].

In this work, we try to address the points highlighted above in the context of fast online search and parameter estimation for MMA with GWs. We report `AMPLFI`[11], a PE algorithm based on LFI using normalizing flows. Though it can be run standalone, the primary focus of `AMPLFI` is to run alongside neural-network based CBC search `Aframe` [31], and compute GW alert data products like skymaps and other use source-properties to be sent out with LVK discovery alerts. Though the core principles of LFI and its application are similar to efforts mentioned above, the technical implementation is independent and focused toward online inference. In particular, there are several elements of GW data analysis that are re-implemented as a part of `ml4gw` (codebase: https://github.com/ML4GW/ml4gw), designed for running on accelerated hardware like GPUs for fast and efficient training and inference. Some common set of tools from `ml4gw` are used by both `Aframe` (codebase: https://github.com/ML4GW/aframev2/) and `AMPLFI` (codebase: https://github.com/ML4GW/amplfi), the latter being the focus of this work.

We outline the rest of the paper as follows. In section 2, we motivate our design principles toward running search and PE together. In section 3, we mention optimizations related to data pre-processing and implementing simulations on accelerated hardware which ensure that most of the computation is occurring on the GPU. In section 4, we present the details of a data embedding network which is used to summarize the data. This embedding is pre-trained using a self-supervised method to create data summary, marginalizing parameters like time of arrival that are reported by the search. In section 5, we give the details of our normalizing flow implementation. We present results and benchmarks in section 6, before concluding in section 7.

[10] This is done in case of stochastic signals offline, for example, see [28].

[11] **A**ccelerated **M**ultimessenger **P**arameter estimation using **LFI**; pronounced 'amp-li-fy'.

2. Aframe + AMPLFI

In order to build as fast of a system as we can, we have made a number of design choices when building the Aframe + AMPLFI framework that we highlight below:

- A modular design to perform search and PE. The search for GW signals in this case is done by Aframe. Candidates from Aframe provide an estimate of the time of arrival and the significance via a false-alarm-rate. This is unlike traditional match-filtering searches that provide, in addition, the best matching template, and the corresponding SNR time series that is used by other annotation algorithms to provide skymaps [29] and source properties [30] of binary systems. In the proposed framework, once a segment of data is found to be of high-significance i.e. containing a GW signal, the parameters are inferred using AMPLFI. Hence, Bayesian parameter estimation results are available along with the discovery of the candidate.
- Data is held in GPU memory to minimize overheads in communication between different components in the low-latency alert infrastructure [4]. For example, Aframe runs as a service, maintaining a buffer[12] of the data in GPU memory. Once a trigger occurs, the relevant segment is passed to AMPLFI for inference. Based on model size of Aframe and AMPLFI, both are able to be served on a single GPU like NVIDIA A30. This reduces any inference overheads as the data is kept on the same device. However, the models may be served as separate micro-services in case the model size or running on a single GPU turns out to be a barrier.
- The accuracy of the results are suited for 'online' purposes i.e. suitable for follow-up efforts, but not necessarily the most refined. The aim is to provide data products like skymaps and source properties for real time discovery alerts. Therefore, we restrict ourselves to GW waveform models that capture the inspiral-merger-ringdown phases, but do not focus on physics of higher-modes, spin precession etc and prioritize fast inference for data products required for follow-up.

For AMPLFI we use a normalizing flow to learn the posterior distribution directly using simulations of binary black hole signals (BBHs). We also make some optimizations compared to previous efforts in light of an online inference algorithm:

- We use real detector data from the LIGO GW instruments during training. Most previous efforts in LFI use simulated, colored Gaussian noise (see, for example, [27, 32]).
- We use efficient data loading and whitening tools to minimize the data transfers back and forth between CPU and GPUs (or other accelerators). We elaborate this below in section 3.
- We re-implement CBC waveform generation on the GPU memory to directly generate waveforms on-the-fly during training.

3. Simulations on Accelerated Hardware

3.1. Waveform model

We use the IMRPhenomD phenomenological waveform model for our simulations [33]. This waveform model contains the full inspiral-merger-ringdown physics, starting with the inspiral phase up to 3.5 post-newtonian order in GW phase (known as TaylorF2; see [34] for a review), and using an ansatz for the merger and ringdown, fitting them to numerical relativity results. One limitation of this waveform model is the restriction to aligned spins i.e. BH spin components perpendicular to the orbital plane and therefore no precession. Current online PE using stochastic sampling techniques use the IMRPhenomPv2 waveform approximant, which contains precessing effects. Furthermore, high mass BBH systems use the IMRPhenomXPHM approximant, which also includes higher modes of radiation. However, we note that inference like sky-localization is insensitive to such effects. Also EM-brightness of a binary depends primarily on the aligned spin components aside from the mass ratio. Hence, the use of aligned-spin is justified for online purposes. In the future, however, we plan to implement and integrate the IMRPhenomPv2 waveform model with our workflow.

In figure 1, we show the time-domain strain of a representative BBH system based on our IMRPhenomD implementation and compare it with that implemented in lalsimulation. The latter is a part of the LIGO Algorithm Library [35], and provides the core components of GW data analysis with LVK data. We find consistency between our implementation and that in lalsimulation, with the residual errors below the signal by three orders of magnitude through most of the evolution. Such residuals are unlikely to impact the results since the statistical errors of posterior is greater than systematic error due to

[12] A snapshotter that only sends new data segments into GPU memory.

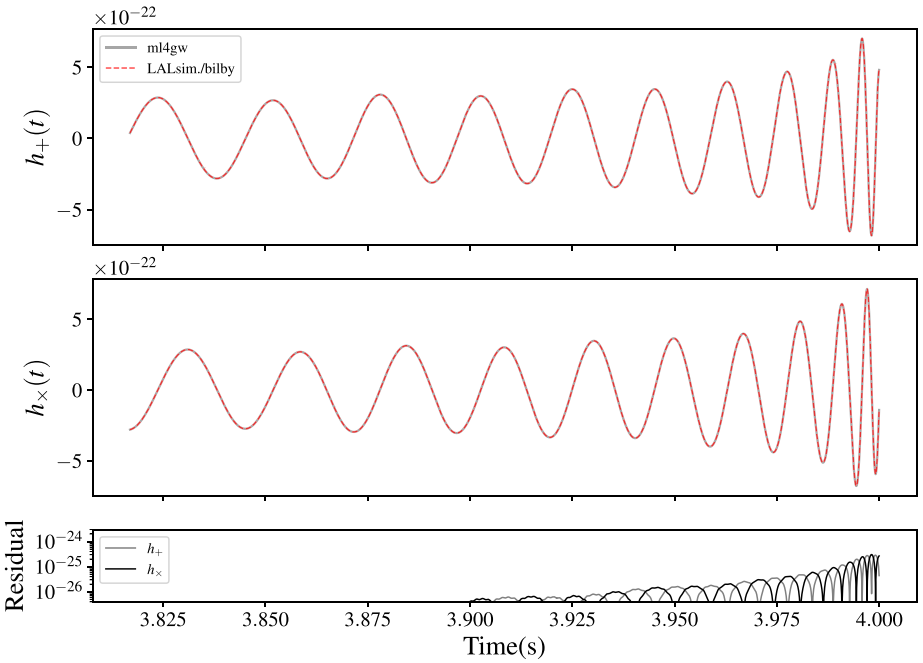

Figure 1. A comparison of time-domain `IMRPhenomD` between that implemented in this study, as a part of the `ml4gw` library. The parameters of the waveform is $\mathcal{M} = 26\, M_\odot, q = 1.0, D_\mathrm{L} = 1000$ Mpc, $\chi_{1,2} = 0.0$. We find that while there are differences in the waveform strain, the residuals are below three orders of magnitude compared to the signal for most of the evolution, except the final few cycles where it is two orders of magnitude lower.

such differences. We present some comparison results in section 6. Though we show comparison with a single representative system in figure 1, several other combination of parameters are tested for consistency with `lalsimulation` as a part of unit-tests of the `ml4gw` codebase.[13]

3.2. Data generation on the GPU

Generally, neural-network models are trained using batches (also called mini-batches) of data that is pre-processed on the CPU and then transferred to the GPU (or other co-processor) to carry out the forward/backward passes, and updating the model weights. However, this may leave the GPU under utilized if the pre-processing and data transfer between the CPU/GPU takes greater time compared to the operations to train the model. This is especially important in the context of LFI since efficient training relies on providing unique combinations of parameters and data to approximate the distribution. We therefore take a different approach by performing the data generation on the GPU, which allows generation of batches of data directly on the device, which is faster and avoids the data transfer overheads. Additionally, all data pre-processing, like fourier transforms, power spectra estimation, data whitening, are carried out on the GPU. This ensures consistent high GPU utilization. Also, we can take advantage of the fact that GPU architecture today provide large memory to generate large batches of data. Our workflow involves:

- Transferring a chunk of two detector (Hanford and Livingston, subsequently HL) time-domain strain data, typically quarter of a day, sampled at 2048 Hz to the GPU before commencing training. The power spectra is fit to this background chunk. During training, N small background chunks, each of 4 s duration, are lazily loaded from the total training chunk, where N is the training batch size.
- We generate N points from our parameter prior and generate the `IMRPhenomD` waveforms directly on the GPU, as mentioned in section 3.1. This step is fast, for example, generating $N = 1000$ waveforms $\sim$0.15 s on a NVIDIA A40 GPU.
- We then inject the signals into the background chunks, obtaining the data batch. We whiten the batch using the estimated power spectra and pass it along with the parameters for training/validation/testing. Examples of the whitened data with the injected waveform overlayed is shown in the panels of figure 2.

[13] found at https://github.com/ML4GW/ml4gw/blob/v0.5.0/tests/waveforms/test_cbc_waveforms.py.

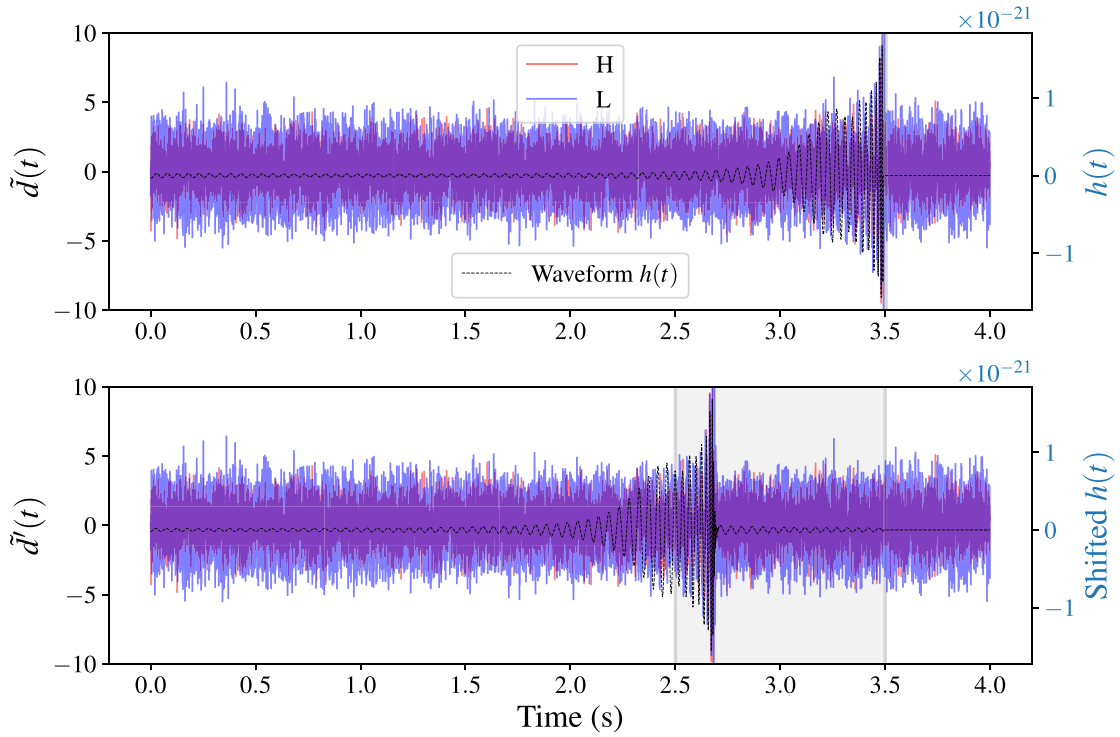

Figure 2. The figure shows the whitened time-domain background strain from Hanford and Livingston (HL) in two different colors. This is a stretch of data from May 2019 (early O3). A simulated BBH waveform, $h(t)$, is injected, shown by the dashed curve. The left y-axis denotes the range of the whitened data after injection, while the right y-axis denotes the waveform strain $h(t)$. In the bottom panel, we see the same background strain with a time-shifted waveform injected, the parameters of which are otherwise the same as the top panel. The time shift is random up to 1 s compared to the top panel, indicated by the shading. We summarize our data views like d and d', and embed them jointly. Subsequently, for LFI, we only use d'-like views.

We call this implementation `InMemoryDataset` in `ml4gw`. We also note that though we have used GPUs as the co-processors in this work, our software framework can also be ported to other accelerators, like TPUs or HPUs, supported by the `pytorch-lightning` [36] framework that we use.

4. Embedding Network

The input to the neural network model is a 2-channel Hanford–Livingston (HL) 4 s whitened time-domain strain. This is projected into a lower-dimensional representation using a embedding network before performing LFI. The coherent analysis of both channels of data is important for some aspects of GW parameter estimation, like sky-localization since it depends on the difference in time of arrival in the different instruments. Our embedding network follows the ResNet architecture used in `Aframe`. The implementation closely resembles that of the `torchvision` library, with some differences.
Firstly, 1-D convolutions are used for time-series, instead of 2D variants used for images. We use group normalization [37] instead of batch-normalization. In `Aframe`, the architecture closely resembles a 34-layer residual network [38]. We, however, avoid the final 512-channel stack of convolution blocks (see figure 3 in [38]) since we do not find performance improvement after including the same. Thus our layer stacks contain blocks of 64, 128, and 256-channel residual convolution blocks. The number of convolution layers in each block is determined by hyper-parameter optimization (HPO) using Variance-Invariance-Covariance Regularization (VICReg) [39] loss, detailed below in section 4.1. Details about the HPO are presented in appendix A. The best configuration resembles an analogous 24-layer ResNet. A final fully-connected layer projects to a representation dimension, D_γ, also determined as a part of the HPO.

4.1. Self-supervised learning of nuisance parameters

We pre-train the embedding network to marginalize over uncertainties in arrival time up to 1 s. This is done since the *peak* of the detection statistic reported by `Aframe` may differ from the true arrival time up to tens of milliseconds. We choose 1 s as a conservative upper bound for the same. The pre-training is done via self-supervised learning (SSL) by identifying two 'views' of the data as being the similar, and training the embedding network to minimize VICReg. Examples of two different views, d and d', are shown in figure 2, where the upper panel shows a signal that is injected at a fixed reference time, while the lower panel shows a

time-shifted signal i.e. all signal parameters being the same except the time of arrival, which is chosen randomly up to 1 s in this case. We should emphasize that the views shown in figure 2 do not affect the relative times of arrival between the detectors which is crucial for sky-localization, but instead are a shift in the geo-center time of arrival by up to 1 s. Two batches of views are then forward-modeled through the embedding network and projected down to the resulting space. This projection, $\mathbf{\Gamma}$, is performed in two different steps. First, the ResNet f, mentioned above, projects the inputs in to $\boldsymbol{\gamma} \in \mathbb{R}^{D_\gamma}$. Following this, another fully-connected expander network, h, takes γ to a $(N \times D_\gamma)$-dimensional space $x \in \mathbb{R}^{N \times D_\gamma}$. The resulting composition is given by $\mathbf{\Gamma} \equiv h \circ f$,

$$\boldsymbol{\gamma} = f(\boldsymbol{d})\,;\; \boldsymbol{\gamma}' = f(\boldsymbol{d}')\,;\; x = h(\boldsymbol{\gamma})\,;\; x' = h(\boldsymbol{\gamma}')\,. \tag{1}$$

We follow the prescription mentioned in [39] and compute the $\mathcal{L}_{\rm VICReg}$ loss in the expanded dimension as,

$$\begin{aligned}\mathcal{L}_{\rm VICReg}(x, x') = \lambda_1\, {\rm MSE}(x, x') + \lambda_2\, \left[\sqrt{{\rm Var}(x) + \epsilon} + \sqrt{{\rm Var}(x') + \epsilon}\right] \\ + \lambda_3\, \left[C(x) + C(x')\right].\end{aligned} \tag{2}$$

Here, MSE is the mean-squared error between the two projected views. The second term involves the variances of the individual batches, regularized by a tolerance to prevent collapsing to zero. Finally, the third term is the quadrature sum of the off-diagonal entries in the individual covariance matrices of the views. The $\lambda_{1,2,3}$ are relative weights of each term, which is also tuned as a part of HPO. We would like to note that the expander network, h, is only required for this pre-training step, and is not required for the subsequent posterior estimation step.

Previous work using LFI reported other techniques, like group equivariance, to tackle such symmetries [40]. We, however, take a different approach since parameters like time of coalescence as reported by the search, despite having some uncertainty associated, is sufficient for follow-up. Hence, we marginalize over it and perform inference on parameters like masses and sky location which are also needed for follow-up. For more details on this technique, the reader is referred to [41]. This technique can be extended to other nuisance parameters in case of GWs, for example the coalescence phase. This is left to future work.

5. Posterior Estimation

Posterior estimation in LFI involves learning the posterior, $p(\mathbf{\Theta}|\boldsymbol{d})$, using an approximator, $q_\varphi(\mathbf{\Theta}|\boldsymbol{d})$, using simulations $\{\mathbf{\Theta}_i, \boldsymbol{d}_i\}$. The parameters φ are adjusted to maximize the likelihood of the simulations, which is mathematically equivalent to minimizing the Kullback-Leibler (KL) divergence between the true posterior and the approximator. The loss function used is,

$$-\ln \mathcal{L}(\varphi) = -\frac{1}{N_{\rm sims.}} \sum_{i \in {\rm sims.}} \ln q_\varphi(\mathbf{\Theta}_i|\boldsymbol{d}_i)\,, \tag{3}$$

where the simulations are forward modeled to calculate their likelihood, which is then maximized during training. The density evaluations are done by learning a set of variable transforms that take the original variables $\mathbf{\Theta}$ to variables of a simpler base distribution, like a standard normal, which we use here. Several techniques are used to build flexible transforms and preserve the probability density at each stage. We refer the reader to a review article on normalizing flows and the different implementations [42]. In our case the parameter space is 8-dimensional,

$$\mathbf{\Theta} = \{\mathcal{M}, q, D_{\rm L}, \theta_{\rm JN}, \alpha, \delta, \phi_c, \psi\}\,. \tag{4}$$

Here, $\mathcal{M} = (m_1 m_2)^{3/5}/(m_1 + m_2)^{1/5}$ is the chirp mass of the binary, $q = m_2/m_1$ is the mass ratio of the binary defined to be less than unity, $D_{\rm L}$ is the luminosity distance of the source, $\theta_{\rm JN}$ is the inclination of the orbit w.r.t. the line-of-sight, α and δ are right ascension (RA) and declination respectively, ϕ_c is the coalescence phase, and ψ is the polarization angle. The prior distribution used for generating the simulations is mentioned in table 1. Note that the time of coalescence is not a part of parameter set since it is marginalized over. Although the `IMRPhenomD` supports BH spins, we have ignored it for the current work. This will be relaxed in subsequent versions of the code and reported in a future work. The data, $\mathbf{d}$, consists of 4 s of whitened time-domain strain sampled at 2048 Hz containing a BBH signal, as illustrated in figure 2. When training the normalizing flow, we condition the parameters on the data representation, $\boldsymbol{\gamma}$, as shown equation (1). This uses only the ResNet, f. The expander, h, is not required subsequently. Also, we leave the

Table 1. Prior distributions of parameters. Note that the distance prior is a power-law with index 2 i.e. uniform in volume; cosmological effects are not included.

Parameter	Prior
$\mathcal{M}$ (Chirp mass)	Uniform(10, 100)$M_\odot$
q (Mass ratio)	Uniform(0.125, 1)
D_L (Lumin. dist.)	Uniform in Vol.(100, 3000) Mpc ($\sim {D_\mathrm{L}}^2$)
θ_JN (Inclination)	Sine(0, π)
α (RA)	Uniform(0, 2π)
δ (Dec.)	Cosine($-\pi/2$, $\pi/2$)
ϕ_c (Coal. phase)	Uniform(0, 2π)
ψ (Pol. angle)	Uniform(0, π)

weights of f to change further as a part of training the normalizing flow. Hence, our normalizing flow maximizes,

$$-\ln \mathcal{L}(\varphi) = -\frac{1}{N_\mathrm{sims.}} \sum_{i \in \mathrm{sims.}} \ln q_\varphi \left(\boldsymbol{\Theta}_i | f(\boldsymbol{d}_i)\right), \tag{5}$$

where the difference between equation (3) vs. equation (5) is the conditioning on the data summary (see equation (1)), pre-trained to marginalize time of arrival. It should be mentioned that the pre-training step with VICReg loss is significantly cheaper compared to training the normalizing flow. In our experiments, we found the pre-training requiring few-tens of epochs of training, which took less than an hour to reach early-stopping condition on a NVIDIA A40 GPU.

5.1. Autoregressive Flows for LFI

Our normalizing flow implementation uses inverse auto-regressive transforms [43]. This kind of auto-regressive transforms can be sampled in one forward pass. However, evaluating the density requires D-forward passes, where D is the dimensionality of the parameter space i.e. $D = 8$ from equation (4). Masked auto-regressive transforms [44] on the other hand use similar masked linear layers [45], but on the contrary the density evaluation takes a single forward pass and sampling is D-times as expensive. Although auto-regressive flows are *universal approximators* [42], our choice of using inverse auto-regressive flow (IAF) is because our inference requires fast sampling, which is achieved in a single pass with the IAF. In addition to IAF, coupling transforms were also considered. However, in our experiments, we found such transforms to perform less optimally when constrained to the same number of trainable parameters.

Our transforms are implemented using the open source library `pyro` [46]. The complete transform is composed of 60 individual affine-autoregressive transforms. Each transform has 6 masked linear layers; each layer having 100 hidden units. More complex transform functions like monotonic splines and neural-network with positive weight exist in the literature. However, we use the affine transforms for simplicity and lower number of trainable parameters. We train the network with a batch size of 800, with 200 batches per epoch using the AdamW optimizer [47] with initial learning rate of 1×10^{-3} and weight decay of 2×10^{-3}. The learning rate is scheduled to reduce by a factor of 10 upon plateauing of the validation loss with a patience of 10 epochs. These configurations are decided after hyper-parameter tuning over a combination of several hundred parameter combinations detailed in B. Our trainer is scheduled to terminate training once the validation loss saturates with a patience of 50 epochs. In terms of training time, we find training $\gtrsim$200 epochs with the above configuration takes $\sim 20 - 24$ h depending on a single 40GB NVIDIA A40/40GB NVIDIA A100 GPU. We note that because our dataset is generated on-the-fly, distributed training does not provide any benefit across multiple devices in terms of training time, however, the training sees more data by a factor of the number of devices used. We, however, did not find significant differences in model performance by training across one vs. multiple devices. In terms of number of trainable parameters, the embedding network contains $\sim$2.6 million parameters, while the auto-regressive transforms contain $\sim$3.2 million parameters, totaling to $\sim$6 million parameters. We would like to note that with on-the-fly data generation on the GPU, a typical training run sees $\gtrsim$200 epochs $\times$200 batches per epoch $\times$800 batchsize $\sim$32 million unique simulations. This implies that unlike most 'large' neural-networks in the literature today, our network is not over-specified in the sense that training data samples exceed the number of trainable parameters by about a factor of five.

In terms of inference, average sampling time for drawing 20 000 posterior samples, conditioned on new data, takes $\sim$0.05 s on NVIDIA A40 GPU. The same on a Intel Core i7 with 16 cores, takes $\sim 1 - 2$ s. However, it should be noted to create a sky-localization map in the HEALPix format [48], used in the GW data analysis, the density needs to be evaluated across all pixel coordinates. We find the average time to

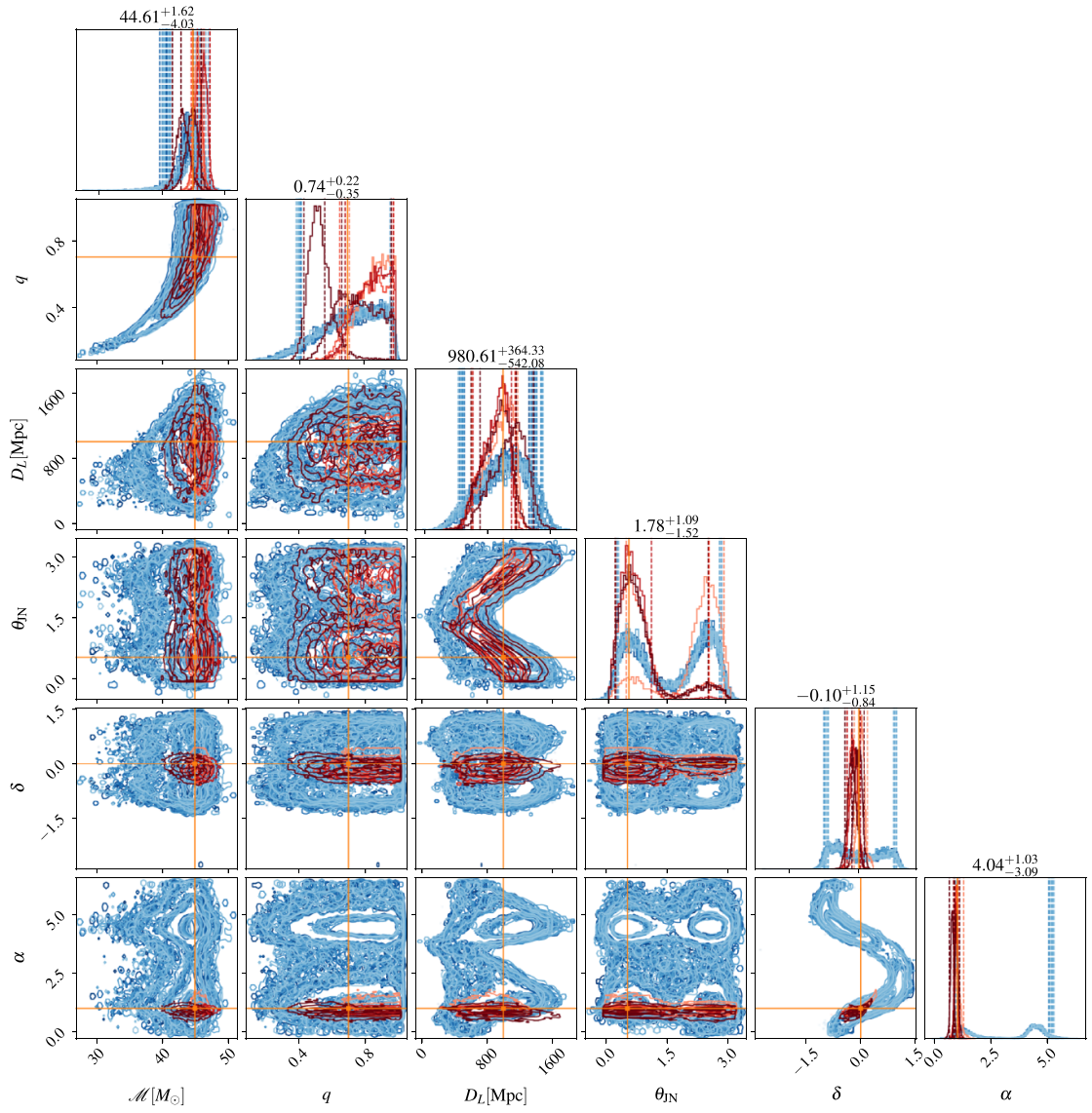

Figure 3. Example posterior for a signal with parameters $\{\mathcal{M} = 45M_\odot, q = 0.7, D_L = 1000$ Mpc, $\theta_{JN} = 30$ deg., $\alpha = 1$ rad., $\delta = 0$ rad.$\}$ injected in 20 different background instances, sampled using `AMPLFI` is shown in sky blue . All posteriors are consistent with one another. Posteriors from the same signal injected in 5 different background instances (same background stretch as the `AMPLFI` injections) and analyzed via nested sampling with Bilby, is overlayed in varying Orange-red colors. The values mentioned above each marginal distribution correspond to the `AMPLFI` median with the 5th and 95th percentiles. We pick one out of the 20 cases for the values mentioned since there is overall agreement amongst them. We find that parameters like $\mathcal{M}$ and q are consistent in terms of detection uncertainties across different runs. Extrinsic parameters, especially the sky-location, though consistent with the true parameters, shows larger uncertainty with `AMPLFI`.

drawn 20 K samples, and then evaluate the density across all pixels on the sky for a HEALPix resolution of NSIDE $= 32$ is $\sim$0.6 s. This is due to the choice of the inverse auto-regressive flow, sampling is possible via one forward pass, but evaluating the density is done sequentially across each component. Doubling the resolution, i.e. using NSIDE $= 64$ takes $\sim$1.2 s and NSIDE $= 128$ takes $\sim$2.4 s.

6. Results and Performance

We show example posteriors in figures 3–5 from representative higher and lower mass BBH source under different conditions. In all cases, the signal injection is performed in a background different from that used during training. The same signal is injected 20 times at different background segments and samples are drawn. The posteriors are shown in the blue colormap. We also perform inference on the same system via nested sampling using `Bilby` [49, 50] and `Dynesty` [51] with 1500 live points and identical priors as when training `AMPLFI`, repeating 5-times on different background segments. This is shown in the red colormap. We see that there is consistency amongst the `AMPLFI` posteriors i.e. distributions in blue colormap fall on top of each other. For higher masses, like the $\mathcal{M} = 45\ M_\odot$, shown in figure 3, we find the recovery accuracy of

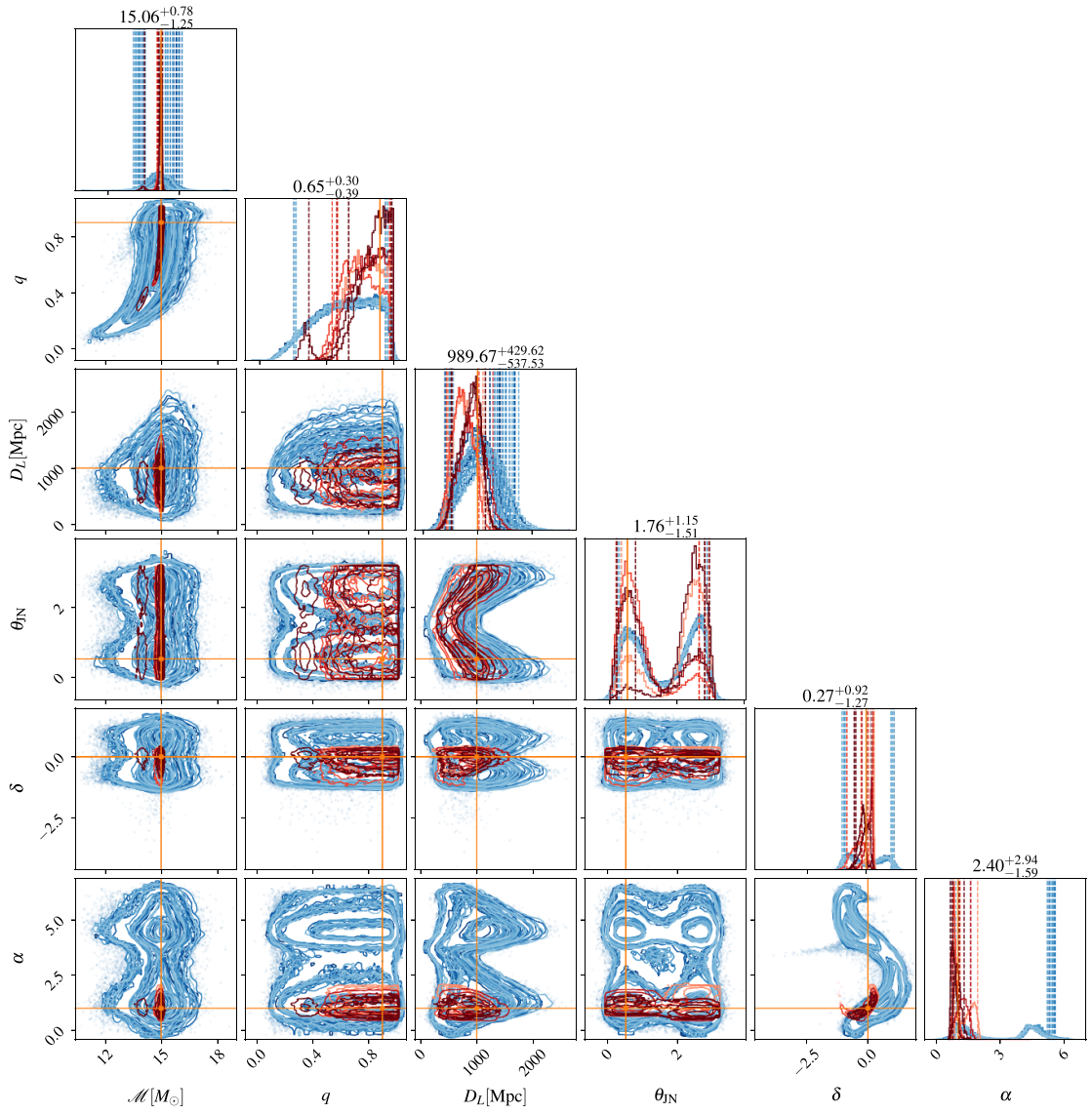

Figure 4. Example posterior for a signal with parameters $\{\mathcal{M} = 15M_\odot, q = 0.9, D_L = 1000$ Mpc, $\theta_{JN} = 30$ deg., $\alpha = 1$ rad., $\delta = 0$ rad.$\}$ injected in 20 different background instances, sampled using `AMPLFI` is shown in sky blue. All posteriors are consistent with one another. Posteriors from the same signal injected in 5 different background instances (same background stretch as the `AMPLFI` injections) and analyzed via nested sampling with Bilby, is overlayed in varied Orange-red colors. The values mentioned above each marginal distribution correspond to the `AMPLFI` median value with the 5th and 95th percentiles. We pick one out of the 20 cases since there is overall agreement amongst them. We find that parameters like $\mathcal{M}$ and q are consistent in terms of detection uncertainties across different runs. Extrinsic parameters, especially the sky-location, though consistent with the true parameters, shows larger uncertainty with `AMPLFI`.

the chirp mass to be comparable to nested sampling. Also, considering all nested sampling runs, the mass-ratio accuracy from `AMPLFI` is comparable. However, extrinsic parameters like the inclination, or the sky location are not recovered with the same accuracy as nested sampling, although broad features like the 'ring' pattern in the skymap are evident. The same is also true for the inclination posterior, where the inference is degenerate between the true inclination angle, and its supplementary angle. In case of the nested sampling runs, though this degeneracy is broken, the inference does not always select the right 'peak' for the inclination which can be expected for two-detection observations (see section IV of [52], or [53]). Thus, overall, we find consistency of the `AMPLFI` results with the true parameters of the injection and with nested sampling results, though not as accurate for some parameters.

We find that the inference for lower mass, $\mathcal{M} = 15\ M_\odot$ BBH system, shown in figure 4, is worse compared to higher mass. For example, in figure 4, we see that the $\mathcal{M}$ posteriors recovered by `AMPLFI` is broader compared to that recovered from nested sampling. The extrinsic parameter recovery follows similar pattern as the high-mass example. Though broad features of the sky location like the 'ring' pattern for two detector is evident, the recovery is not at the level of nested sampling results. The greater consistency of the intrinsic parameters compared to the extrinsic parameters suggests one potential avenue of improvement

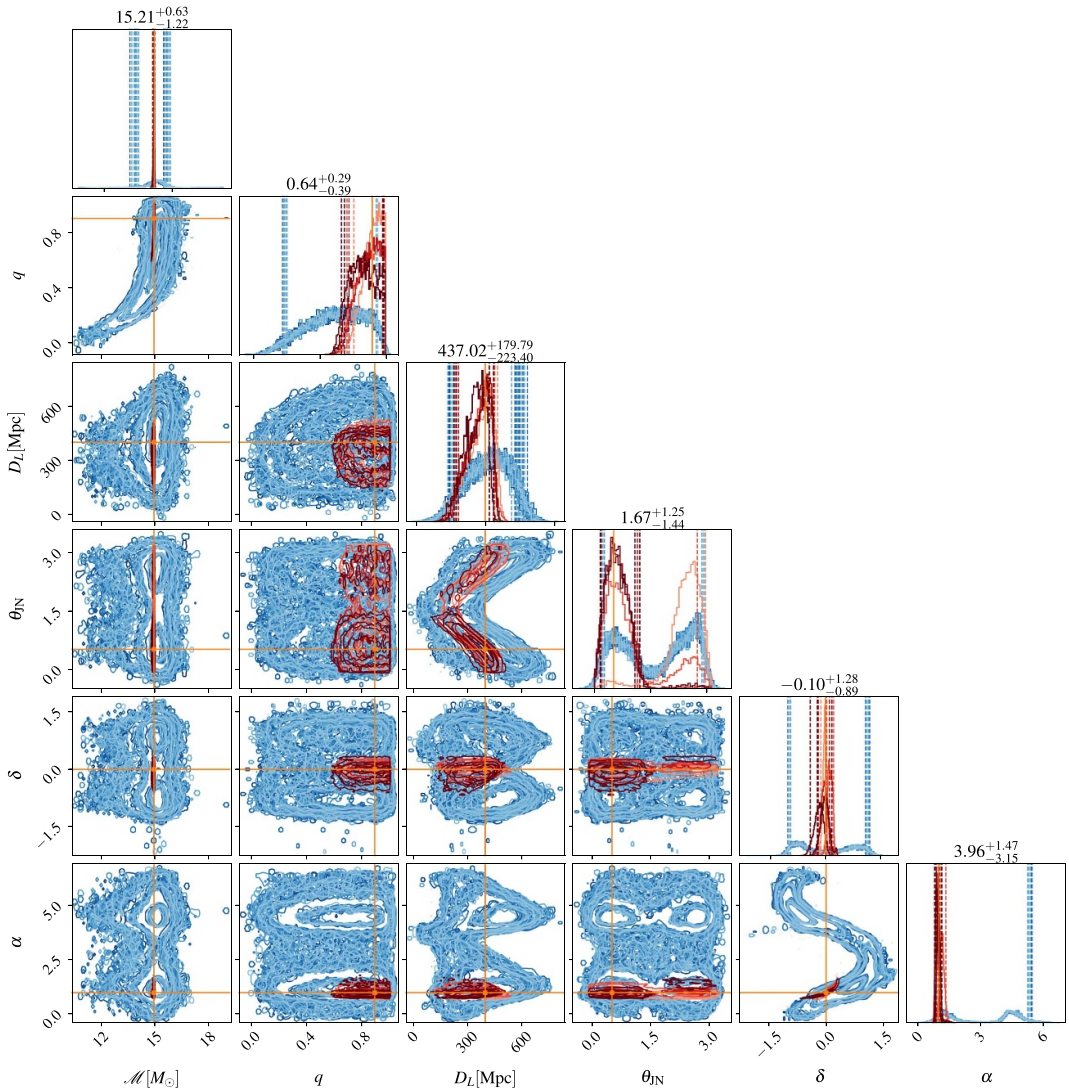

Figure 5. Example posterior for a signal with parameters $\{\mathcal{M} = 15M_\odot, q = 0.9, D_{\rm L} = 400$ Mpc, $\theta_{\rm JN} = 30$ deg., $\alpha = 1$ rad., $\delta = 0$ rad.$\}$ injected in 20 different background instances, sampled using `AMPLFI` is shown in sky blue. All other parameters are kept the same as in figure 4. The choice of $D_{\rm L} = 400$ Mpc ensures that the optimal SNR of this source is similar to that used for figure 3. The analysis with nested sampling in 5 different background instances is performed as in the previous cases and is overlayed in varied Orange-red colors. We find that parameters though the parameter recovery is consistent, the width of $\mathcal{M}$ and q do not change significantly compared to figure 4. However, the $D_{\rm L}$ posterior is more constrained, and closer to the nested sampling posterior.

being to further augment our dataloader in the extrinsic parameters (α, δ, ψ) which is used to project the signal onto the GW antennae. Signal projection as implemented in `ml4gw` performs this operation on-the-fly on the GPU, hence oversampling the corresponding priors and creating a larger batch is feasible without compromising data generation time, and will be considered in a future implementation.

We also compare the two representative systems at similar SNR in figure 5, where the $\mathcal{M} = 15\,M_\odot$ system is simulated at $D_{\rm L} = 400$ Mpc keeping other injection parameters the same as for figure 4. We repeat the analysis done in figures 3 and 4. We find that the widths of the $\mathcal{M}$ and q posteriors do not change significantly when compared to the $D_{\rm L} = 1000$ Mpc injections. This is contrary to the general expectation that the posterior widths scale as $1/\text{SNR}$. However, we find that inference on $D_{\rm L}$ does show that behavior and is more consistent with nested sampling runs. Overall we find consistency among the different `AMPLFI` draws suggesting the algorithms' sanity on small changes to background.

In figure 6, we show parameter recovery consistency with true values via a percentile-percentile (PP) plot for 500 simulated BBHs. The parameters of these are sampled from the same prior as that used during training. Like the previous examples above, for these injections, the background is different from the training background. The diagonal trend of the plot shows that there is no bias in the inference with `AMPLFI` across the parameter space i.e. 10% of the testing data are in 10% credible level, 20% of the testing data in 20%

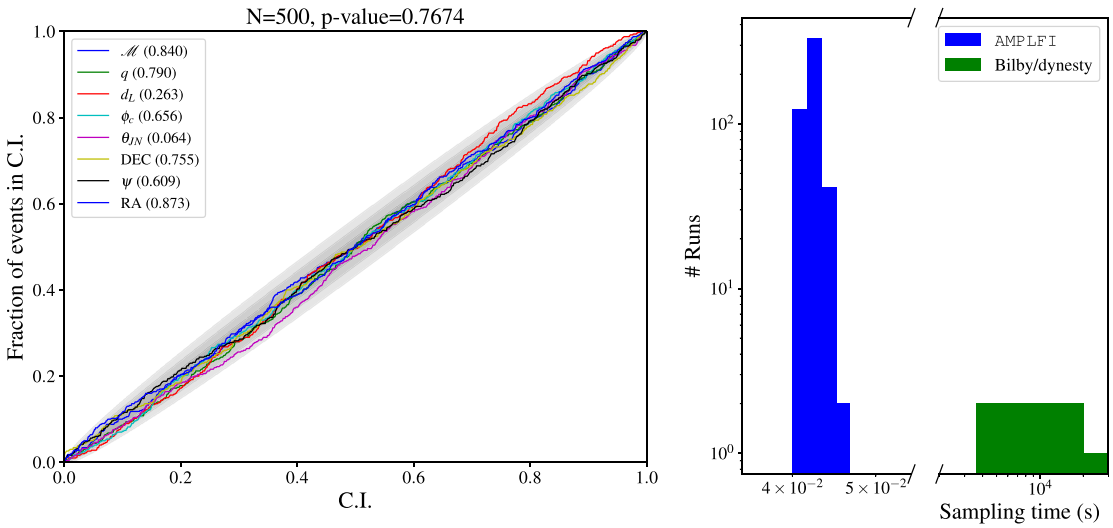

Figure 6. Left: Percentile-percentile (PP) plot showing recovery accuracy for 500 BBH injections performed in a testing background, different from training background. The different lines track the cumulative fraction of events within a corresponding confidence interval for the parameters mentioned in equation (4). The shaded bands represent the 1, 2, 3-σ confidence bands, in decreasing opacity, with respect to a uniform distribution. The mentioned p-values are from a KS test done with respect to a uniform distribution. **Right**: Sampling times for `AMPLFI` vs. nested sampling runs done on injections using Bilby, with identical waveform model and prior settings. The nested sampling runs were done with a CPU pool size of 24, and correspond to the runs using in figure 4. The standard GW likelihood model is used. The `AMPLFI` sampling times correspond to the 500 injections used for the P-P plot on the left.

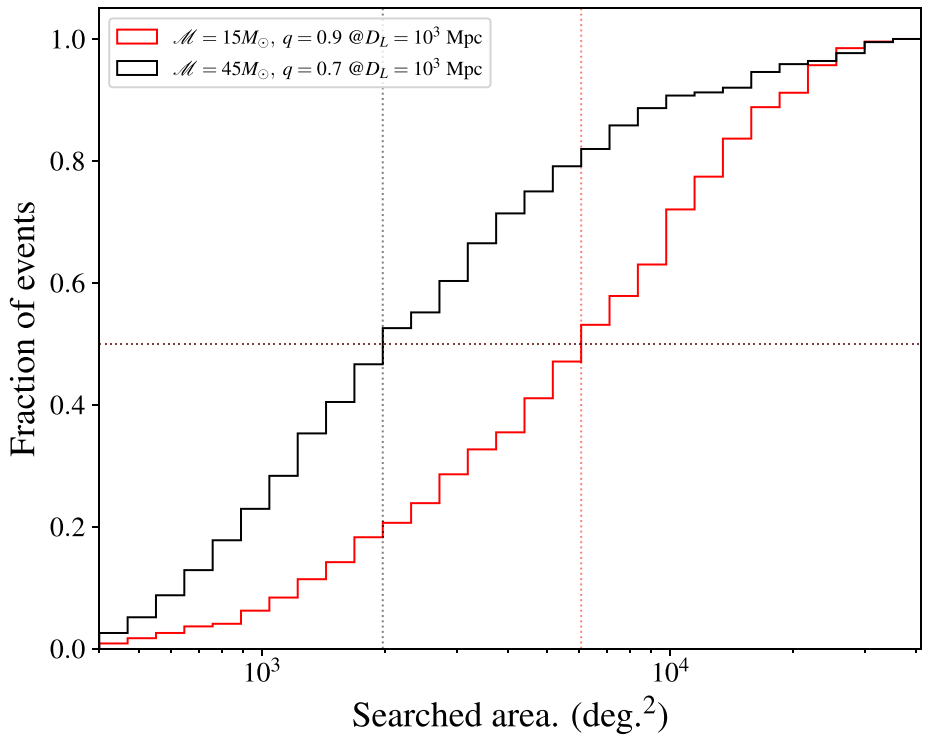

Figure 7. Searched area for a representative system with $\mathcal{M} = 45M_\odot$ and $\mathcal{M} = 15M_\odot$ at $D_\mathrm{L} = 1$ Gpc done across the sky. We note that, as expected, larger chirp mass give better searched area as a result of being louder signals. However, the median searched area is $\mathcal{O}(1000)\mathrm{deg}^2$.

credible level and so on. The shaded regions in the figure represent 1,2,3-σ confidence bands with respect to a uniform distribution. The numbers within parenthesis are p-values from KS test done with respect to a uniform distribution. In figure 7, we show the *searched area* distributions for the two representative higher mass BBH and lower mass BBH placed at a fiducial distance of $D_\mathrm{L} = 1000$ Mpc. The searched area measures the number of pixels between the peak of skymap posterior and the pixel containing the injection in a HEALPix map, when ranked according to the probability density in each pixel. We see the expected trend of

getting a better search area statistic for higher mass system compared to a lower mass system due to their larger amplitude. However, the searched area is several $\mathcal{O}(1000)$ degrees for the two detector model because of the larger error-bars on the sky coordinates mentioned above.

6.1. Comparison with BAYESTAR skymaps

In this section, we run our model on several O3 events using the data from the Gravitational Wave Open Science Center (GWOSC; see https://gwosc.org/) and compare the sky-localization with the corresponding rapid localization method, BAYESTAR [29]. Since our model is trained on only 2-detector (HL) data, we re-run BAYESTAR on the events using only HL SNR timeseries for this comparison. We show the skymap, along with the 90% localization area, for several O3 events that were detected online in figure 8. The left (right) panel shows the reconstruction using `AMPLFI` (BAYESTAR). The choice of the selected events is that their event parameters, as published in GWTC-3, lie within the prior range used by us during training, and the events are spread over the most of O3 from May 2019 to March 2020 (the event identifiers carry the date of discovery). While our training background is a half-day chunk at the start of O3, we test on events which had been detected over the duration of the entire run. This is done to obtain a qualitative assessment of the impact of changing background i.e. if the model performance greatly degrades when supplied with data several months away from the training background. We find that there is broad similarity in the skymaps between `AMPLFI` and BAYESTAR. There is no trend in terms of the skymaps being more/less constrained. However, given that the events used for figure 8 range from May 2019 to March 2020, (recall that training background is limited to half a day in May 2019), we conclude that the model validity is maintained for different background up to several months. This does not imply that the model once trained, may be optimal for the entire run. In fact, one of the requirements for `AMPLFI` is the ability for periodic re-training; however, we anticipate that such re-training may converge quickly given the preliminary observations mentioned here. The suitable re-training cadence is yet to be determined and will be reported in the future, however, we report a preliminary case in the section below.

6.2. Performance on O1 background and GW150914

In this section, we further test the performance of our algorithm on background from the first observing run, O1. We re-train a model using the same architecture and training strategies as in the previous sections except using a 4096-second stretch of background data from September 12, 2015 available in GWOSC. In left panels of figure 9 we show the posterior distributions of $\{\mathcal{M}, q, D_{\rm L}\}$ and the sky-localization when 4-second data around the coalescence of GW150914 [1] is inferred using our model. The solid curve in the corner plot shows the results as reported in GWOSC. The colormap in the mollweide plot shows the sky-localization reported by BAYESTAR for GW150914; the dashed curves in the same plot are the 90% contours from our algorithm. We also 'jitter' the data by a small amount and perform inference, represented by the different colored curves in both the corner plot and the mollweide projection. Specifically, the data used for inference is between $[t_0 - 3\ {\rm s}, t_0 + 1\ {\rm s}]$ with the reported coalescence time of GW150914 being $t_0 = 1126259462.4$. We consider a few representative jitters of t_0 shown in the figure legend, within the 1 s time-shifts used for pre-training the embedding network. This is done to emulate the scenario where the uncertainty in time of coalescence reported by the search does not affect the inference strongly. In particular, for `Aframe` the uncertainty can be upto several tens of milli-seconds, but well within 1-second, which motivates our choice here. We find that all the dotted curves are generally consistent with each other, showing that the data representation is agnostic to small shifts as desired.

For the right panels of figure 9, we consider a background segment from 15 September 2015 and re-train the model starting with the model checkpoint from the above model, we use the identical training configuration except starting with initial learning rate lower by a factor of 10. While the original model early-stopped in $\gtrsim$200 epochs, the latter model achieves the lowest validation loss in in $\sim$40 epochs. We do the inference of GW150914 data in the same way as in the previous case, and find the result mostly consistent, with the sky localization showing some differences but broadly consistent between the two models. We did not use the data from 14 September 2015, since the duration of two-detector data in science mode was minimal due to the poor duty cycle of L1 detector (see analysis ready segments available in GWOSC).

7. Conclusion and Future Work

In this work, we presented a GW parameter estimation algorithm, `AMPLFI`, using likelihood-free inference. This work is one of the efforts to integrate AI algorithms in GW data analysis using tools build as a part of `ml4gw`—like data cleaning using `DeepClean` [54], search of CBCs using `Aframe` [31], anomaly detection algorithm `GWAK` [55]. The use case of `AMPLFI` is to run alongside recently reported neural-network based CBC search, `Aframe`. The intended design is for both `Aframe` + `AMPLFI` to run online, preferably on the

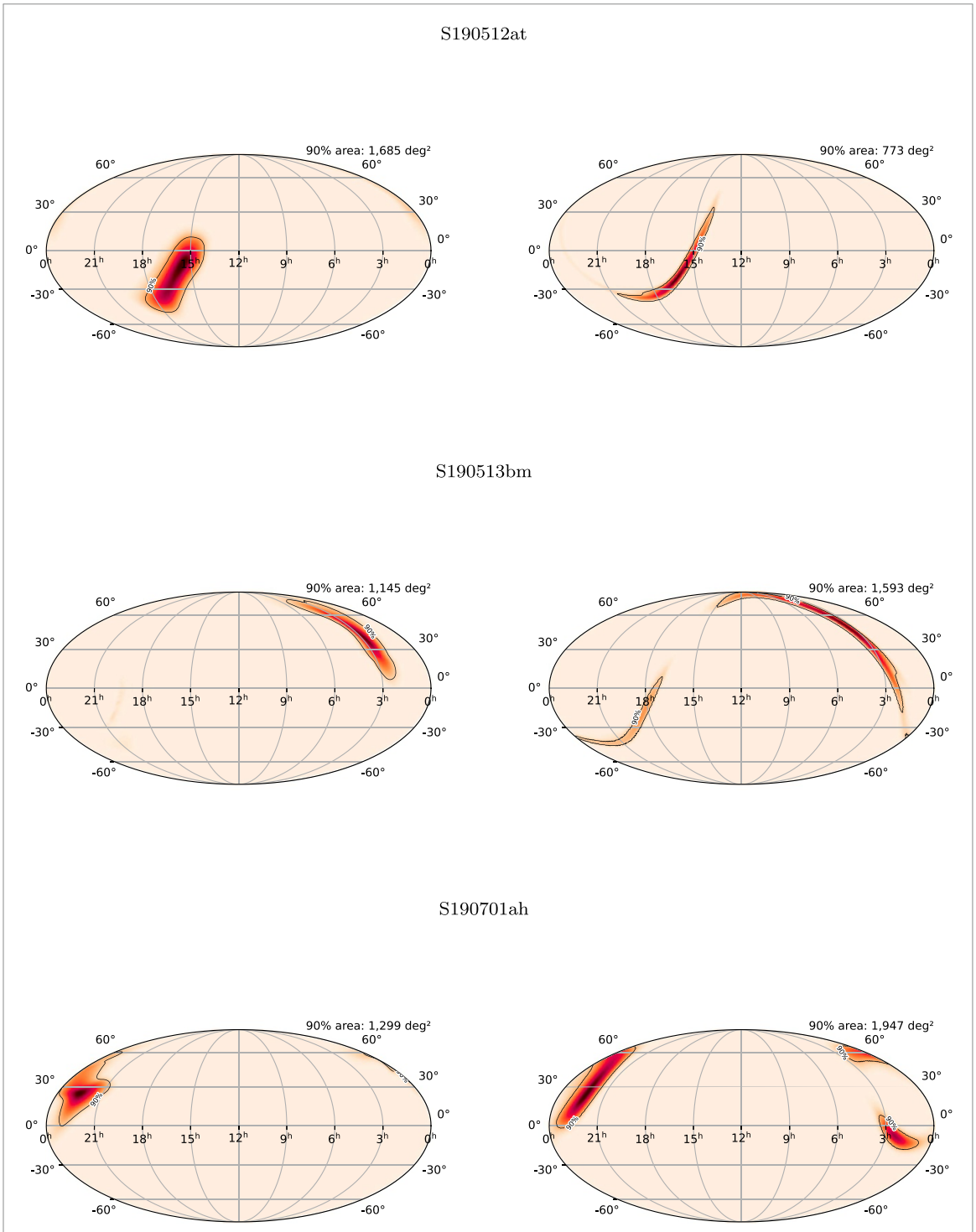

Figure 8. Left: AMPLFI skymaps **Right** Bayestar maps from events, using HL data, spanning different months of the third observing run, O3. We find that the skymaps are broadly consistent, although there is no clear trend of one being more constraining. However, it demonstrates model validity to differing backgrounds up to several months from that used in training.

same hardware to minimize any communication overhead and reduce the time-to-alert. While to date we have focused on this integration of neural-networks in order to achieve very low-latency outputs, there is no reason in principle that AMPLFI could not also be paired with any other search pipeline that provides an estimated time of arrival. An important future product of this research will be a standalone version of the software that can be run simply with the provision of trigger time, enabling easy adoption for searches that want to use it. Current real-time GW alert data products include sky-localization and EM-bright source properties apart from the significance, and a derived data-product, P-astro, based on rate of foreground and background triggers. We have not discussed the EM Bright source properties in this work since the analysis, so far, is limited to BBH signal which are expected to be EM-dark. However, the availability of the posterior

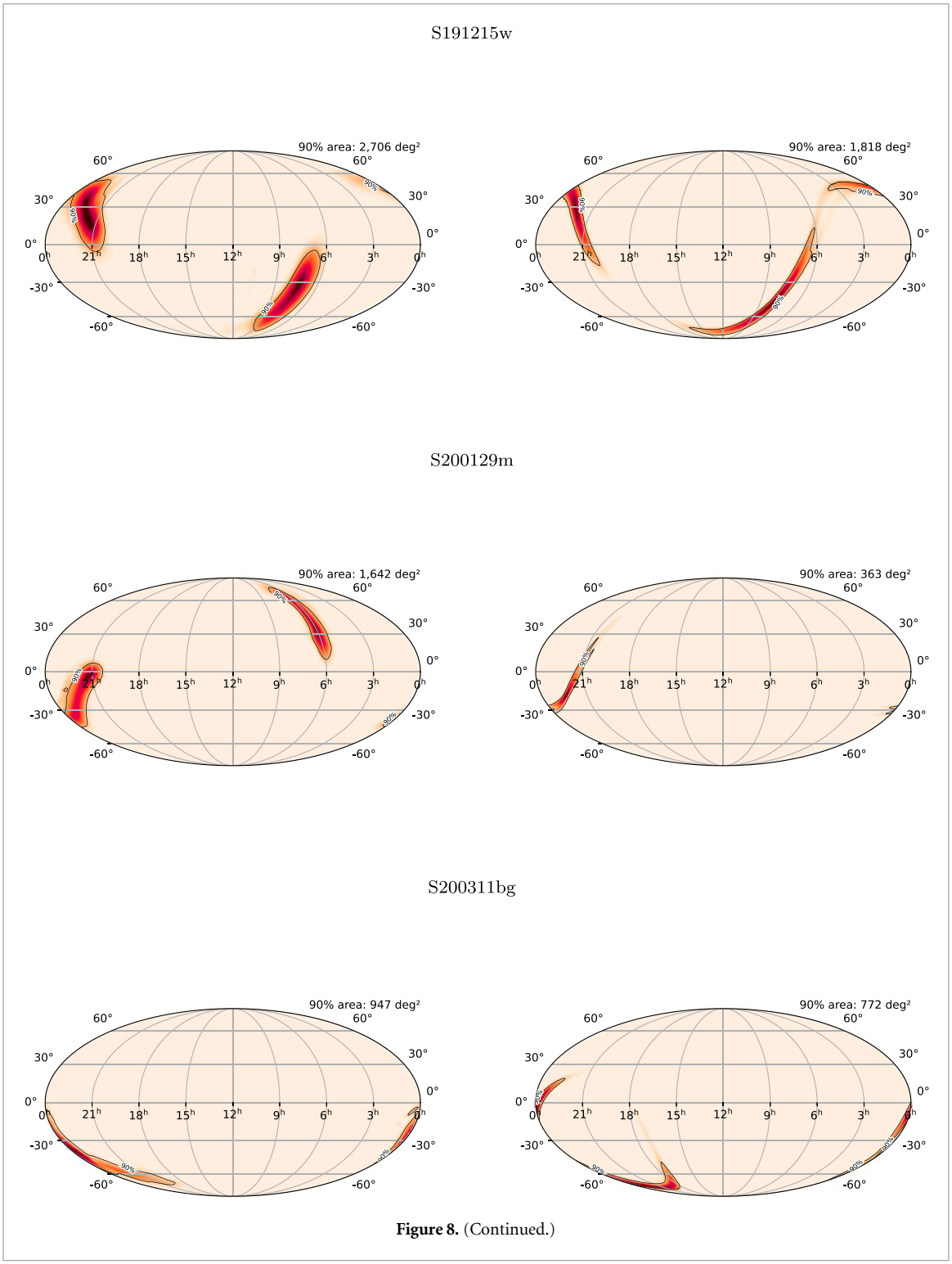

Figure 8. (Continued.)

samples in real-time allows for their straightforward computation by binning the posterior samples, or marginalizing them over several equations of state (see section C.2 of [4]).

As a part of routine end-to-end testing, the LVK has set up a streaming mock data challenge (MDC) with injections over a 40-days chunk of O3 background. This was used to profile latencies in several components in the alert infrastructure, reported in [4]. Preliminary work has been done toward the online deployment of `Aframe` + `AMPLFI` to analyze this MDC. Based on preliminary testing, we find the net latency of data acquisition by `Aframe`, evaluating significance, passing data to `AMPLFI`, followed by generating posteriors and skymaps takes ∼6 s. At the time of writing, candidates are reported to a test instance of GraceDB, the candidate database used by the LVK, however, the view for the same is not public. As a follow-up to the methods reported here, the performance of `Aframe` + `AMPLFI` will be reported on the MDC constructed in [4] in a future work. The MDC dataset contains $\mathcal{O}(1000)$ BBHs, for which several match-filtering searches

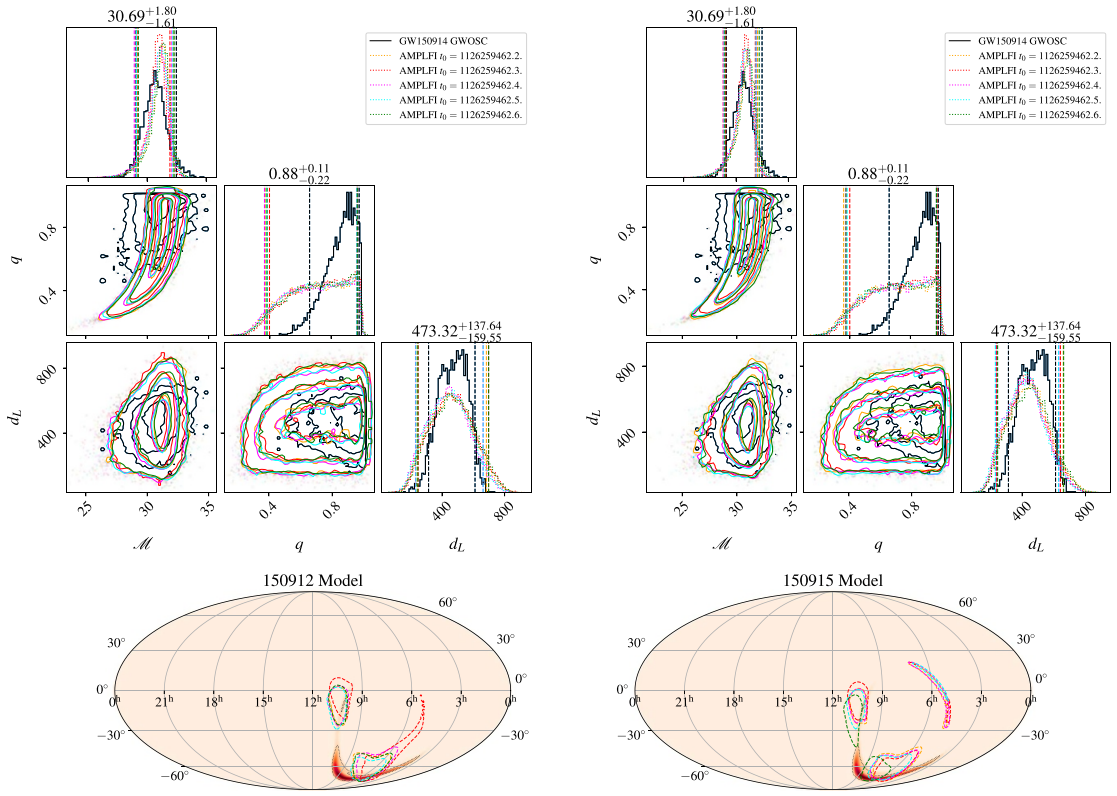

Figure 9. Left panels: The posteriors for $\{\mathcal{M}, q, D_{\rm L}\}$ and sky-localization of GW15019 using an `AMPLFI` model trained on O1 background. The different colors represents 'jitter' in the data segments chosen for inference around the $t_0 = 1126259462.4$ for GW150914. The solid black curve corresponds to the posteriors reported in GWOSC. We find that the results are consistent although with wider confidence intervals, a trend reported in figures 3–5. The consistency of the distributions from `AMPLFI` shows the invariance to small time shifts. **Right panels**: Same as left, except with a model which is retrained with background data from September 15, 2015 starting with the model state from the left panel after training. We find that the results generally agree with the former and that from GWOSC. Some differences in the sky-localization is observed, although the general trend of 'ring' pattern is maintained.

and annotation pipelines including BAYESTAR was run. In the future we plan to conduct a systematic comparison with BAYESTAR on the simulated BBHs in the MDC, along with comparison with online PE results reported in [4].

Certain aspects of the model requires improvements, for example, the inference on the extrinsic parameters like sky location, and the extension to use spinning waveforms. This will be considered in a future work. Also, we note that the focus has been on BBHs due to their shorter signal duration. However, the main focus of MMA is BNS and neutron star black hole systems for which the signal duration can be $\mathcal{O}$(min) depending on the starting frequency. This makes the input arrays larger by an order of magnitude compared to the ones considered here, and therefore expensive in terms of memory and compute. However, low mass systems *inspiral* for most of that duration and the frequency evolution is described analytically, primarily at newtonian order. Thus, feature extraction of the inspiral from the time-series data, or alternative data representations like q-transforms can be a possible approach toward search and parameter estimation.

Finally, the framework for `AMPLFI` is not limited to CBC signals, and can be extended to burst signal morphologies like sine-Gaussians. This is relevant for running parameter estimation on candidates picked up by pipelines like `GWAK` that look for unmodeled events [55]. In this case, a sine-gaussian parameter estimation may lead to measurement of fundamental features like central frequency or duration.

Data availability statement

No new data were created or analysed in this study. The codebase for this work is under development at the AMPLFI repository.

Acknowledgment

The authors acknowledge support from NSF PHY-2117997 'Accelerated AI Algorithms for Data-Driven Discovery (A3D3).' This work used NCSA-Delta at U. Illinois through allocation PHY-240078 from the

Advanced Cyberinfrastructure Coordination Ecosystem: Services & Support (ACCESS) program, which is supported by National Science Foundation Grants #2138259, #2138286, #2138307, #2137603, and #2138296. This work made use of resources provided by subMIT at MIT Physics. This research has made use of data or software obtained from the Gravitational Wave Open Science Center (gwosc.org), [56, 57] a service of the LIGO Scientific Collaboration, the Virgo Collaboration, and KAGRA. This material is based upon work supported by NSF's LIGO Laboratory which is a major facility fully funded by the NSF, as well as the Science and Technology Facilities Council (STFC) of the United Kingdom, the Max-Planck-Society (MPS), and the State of Niedersachsen/Germany for support of the construction of Advanced LIGO and construction and operation of the GEO600 detector. Additional support for Advanced LIGO was provided by the Australian Research Council. Virgo is funded, through the European Gravitational Observatory (EGO), by the French Centre National de Recherche Scientifique (CNRS), the Italian Instituto Nazionale di Fisica Nucleare (INFN) and the Dutch Nikhef, with contributions by institutions from Belgium, Germany, Greece, Hungary, Ireland, Japan, Monaco, Poland, Portugal, Spain. KAGRA is supported by Ministry of Education, Culture, Sports, Science and Technology (MEXT), Japan Society for the Promotion of Science (JSPS) in Japan; National Research Foundation (NRF) and Ministry of Science and ICT (MSIT) in Korea; Academia Sinica (AS) and National Science and Technology Council (NSTC) in Taiwan.

The authors thank Katelyn Wagner for an internal collaboration review of the manuscript. The authors also thank the anonymous referees for helpful feedback.

Appendix A. Hyperparameter tuning of the embedding network

To determine the configuration to be used for the embedding network mentioned in section 4, we perform an extensive hyperparameter optimization over the search space of the layers of ResNet, the convolutional kernel size of the ResNet, the dimensionality of the representation, D_γ, the dimensionality of the expanded space where the VICReg loss is computed; this is tune by a factor, N, i.e. the dimensionality of the expanded representation is $N \times D_\gamma$. We use Stochastic gradient descent optimizer and also sample over the learning rate, the weight decay and the momentum terms of the optimizer. A total of ∼250 training runs were performed using asynchronous hyperbanding with early-stopping [58] using the `ray.tune` library [59]. This technique stops poor performing trials allowing more favorable trials to continue. We use a grace period of 3 epochs before half of ongoing trials are stopped. The experiment was carried over 8 workers over 4 A40 GPU taking ∼40 h. We show the top 10 trial configuration in table A1 and show the epoch-average VICReg validation loss for the same in the left panel of figure A1. The right panel of the figure shows the training/validation of the best model configuration trained until early-stopping condition is met.

Table A1. Top 10 hyper-parameter configurations for hyper-parameter optimization runs for the embedding network. The best trial configuration is shown in boldface.

ResNet conf.	kernel size	λ_1	λ_2	λ_3	D_γ	LR	Momentum	wt. decay	N	VICReg.
(5 , 3 , 3)	**5**	**1**	**1**	**5**	**8**	$\mathbf{7.16 \cdot 10^{-4}}$	$\mathbf{8.07 \cdot 10^{-5}}$	$\mathbf{4.42 \cdot 10^{-4}}$	**3**	**0.48**
(4 , 3 , 3)	7	1	1	1	11	$8.97 \cdot 10^{-4}$	$5.75 \cdot 10^{-5}$	$9.01 \cdot 10^{-5}$	3	0.528
(4 , 5 , 3)	7	1	1	1	9	$2.08 \cdot 10^{-4}$	$2.60 \cdot 10^{-4}$	$3.72 \cdot 10^{-4}$	3	0.539
(4 , 3 , 3)	7	1	1	1	10	$2.62 \cdot 10^{-4}$	$3.69 \cdot 10^{-5}$	$1.16 \cdot 10^{-5}$	3	0.543
(4 , 5 , 3)	5	5	1	1	8	$3.48 \cdot 10^{-4}$	$6.10 \cdot 10^{-4}$	$7.37 \cdot 10^{-4}$	5	0.553
(5 , 5 , 4)	3	5	1	1	8	$1.31 \cdot 10^{-4}$	$1.75 \cdot 10^{-5}$	$1.67 \cdot 10^{-4}$	3	0.578
(3 , 4 , 4)	3	5	1	5	9	$5.38 \cdot 10^{-4}$	$1.42 \cdot 10^{-5}$	$1.67 \cdot 10^{-5}$	5	0.610
(4 , 5 , 3)	3	1	1	5	10	$9.76 \cdot 10^{-4}$	$2.69 \cdot 10^{-4}$	$4.20 \cdot 10^{-5}$	4	0.627
(4 , 3 , 4)	3	1	1	1	9	$1.37 \cdot 10^{-4}$	$5.46 \cdot 10^{-5}$	$1.33 \cdot 10^{-5}$	3	0.677
(4 , 5 , 3)	5	5	5	1	8	$1.64 \cdot 10^{-4}$	$9.47 \cdot 10^{-4}$	$1.61 \cdot 10^{-5}$	3	0.707

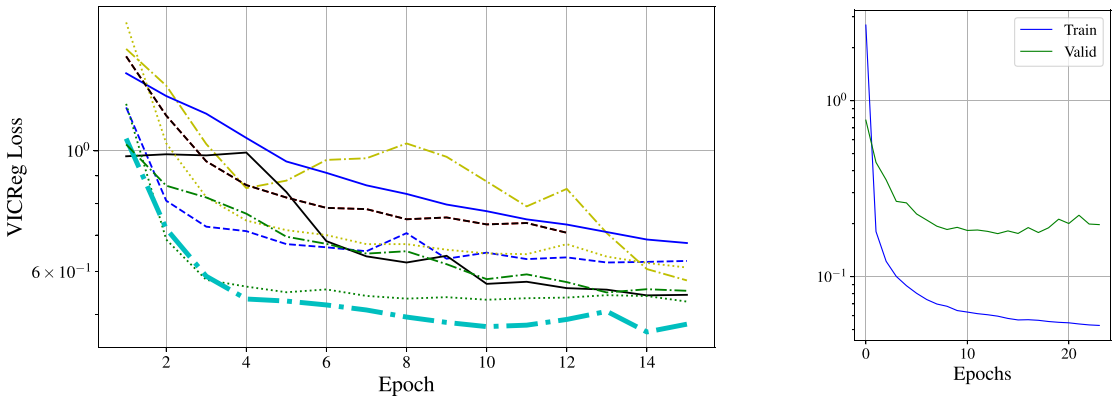

Figure A1. Left: Avg. VICReg validation loss as a function of training epoch from the top 10 HPO runs in table A1. The best configuration is plotted in thick solid line; corresponding configuration is boldfaced entry in table A1. **Right**: Training/Validation loss for best model configuration from the left panel until early-stopping.

Appendix B. Hyperparameter tuning of the normalizing flow

Hyper-parameter optimization is done for ∼100 configurations involving the number of transforms, the configuration of each transform, learning rate, batch size, optimizer weight decay and momentum parameters shown in table B1 using asynchronous hyper-banding with early stopping [58] using the `ray.tune` library [59]. This stops under-performing runs in favor of allowing better performing runs to continue. We carried out the HPO runs over 4 A40 GPUs which took ∼15 h. The average validation loss over validation epoch for the top 5 runs are shown in the left panel of figure B1. The right panel of the figure shows the training/validation loss of the best model configuration trained until early-stopping condition is met. While all the runs were allowed to run for 30 epochs, most of them are stopped early. The validation loss for the top ten performing runs are shown in the figure. We use the topmost configuration in table B1 for the results of the paper. The training and validation for this configuration, trained until early-stopping is shown on right panel of Figure B1.

Table B1. Top 10 hyper-parameter configurations for run involving 30 epochs. Validation loss is noted at the end of the 30th epoch.

# transforms	# blocks	# hidden feat.	LR	batch size	wt. decay	val. loss
60	**6**	**100**	**0.001 29**	**800**	**0.002 41**	**8.52**
100	6	150	0.000 636	1000	0.000 263	8.63
60	6	120	0.002 42	800	0.004 71	8.65
60	8	150	0.000 488	1000	0.000 553	8.70
80	6	120	0.000 442	1000	0.001 61	8.71
60	7	150	0.001 03	1000	0.000 271	8.72
80	7	120	0.000 318	800	0.000 173	8.78
80	6	100	0.000 93	1000	0.003 32	8.83
80	8	100	0.000 873	1200	0.004 53	8.86
60	6	120	0.000 297	1000	0.0509	8.87

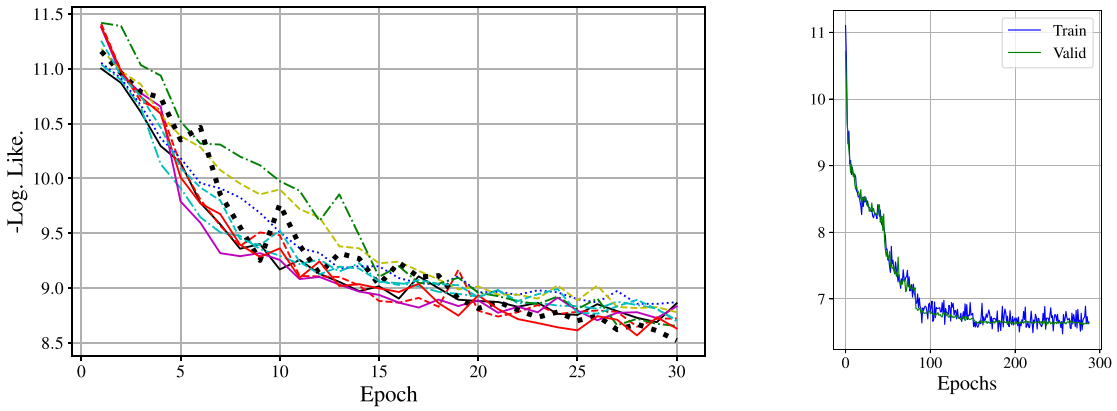

Figure B1. Left: Validation loss from the top 10 HPO runs in table B1. **Right**: Training/Validation loss for best model from the left panel until early-stopping.

ORCID iDs

D Chatterjee https://orcid.org/0000-0003-0038-5468
R Kumar https://orcid.org/0009-0008-6428-7668
M Desai https://orcid.org/0009-0003-4448-3681
E Govorkova https://orcid.org/0000-0003-1920-6618
E Moreno https://orcid.org/0000-0001-5666-3637
R Raikman https://orcid.org/0000-0003-4083-6390
M Saleem https://orcid.org/0000-0002-3836-7751
M W Coughlin https://orcid.org/0000-0002-8262-2924
P Harris https://orcid.org/0000-0001-8189-3741

References

[1] Abbott B P *et al* 2016 Observation of gravitational waves from a binary black hole merger *Phys. Rev. Lett.* **116** 061102
[2] Abbott B P *et al* 2016 GW151226: observation of gravitational waves from a 22-solar-mass binary black hole coalescence *Phys. Rev. Lett.* **116** 241103
[3] Abbott R *et al* 2021 GWTC-3: compact binary coalescences observed by LIGO and virgo during the second part of the third observing run
[4] Sharma Chaudhary S *et al* 2024 Low-latency gravitational wave alert products and their performance at the time of the fourth ligo-virgo-kagra observing run *Proc. Natl Acad. Sci.* **121** e2316474121
[5] Cannon K *et al* 2012 Toward early-warning detection of gravitational waves from compact binary coalescence *Astrophys. J.* **748** 136
[6] Magee R *et al* 2021 First demonstration of early warning gravitational-wave alerts *Astrophys. J. Lett.* **910** L21
[7] Barthelmy S *et al* 2022 Introducing new GCN Kafka broker and web site for transient alerts, https://gcn.nasa.gov *GRB Coordinates Netw.* **32419** 1
[8] Wyatt S, Tohuvavohu A, Arcavi I, Sand D, Lundquist M Howell D A, McCully C and Riba A, Burke J and Gravitational Wave Treasure Map Team 2019 Announcing the GW treasure map *GRB Coordinates Netw.* **26244** 1
[9] van der Walt S J, Crellin-Quick A and Bloom J S 2019 SkyPortal: an astronomical data platform *J. Open Source Softw.* **4** 1247
[10] Street R A, Bowman M, Saunders E S and Boroson T 2018 General-purpose software for managing astronomical observing programs in the LSST era *Proc. SPIE* **10707** 1070711
[11] Buikema A *et al* 2020 Sensitivity and performance of the advanced LIGO detectors in the third observing run *Phys. Rev.* D **102** 062003
[12] Aasi J *et al* 2015 Advanced LIGO *Class. Quantum Grav.* **32** 074001
[13] Acernese F *et al* 2015 Advanced virgo: a second-generation interferometric gravitational wave detector *Class. Quantum Grav.* **32** 024001
[14] Akutsu T *et al* 2020 Overview of KAGRA: Detector design and construction history *Prog. Theor. Exp. Phys.* **2021** 05A101
[15] Abbott B P *et al* 2017 GW170817: Observation of gravitational waves from a binary neutron star inspiral *Phys. Rev. Lett.* **119** 161101
[16] Abbott B P *et al* 2017 Multi-messenger Observations of a binary neutron star merger *Astrophys. J. Lett.* **848** L12
[17] Coughlin M W 2020 Lessons from counterpart searches in LIGO and virgo's third observing campaign *Nat. Astron.* **4** 550–2
[18] Canizares P, Field S E, Gair J, Raymond V, Smith R and Tiglio M 2015 Accelerated gravitational-wave parameter estimation with reduced order modeling *Phys. Rev. Lett.* **114** 071104
[19] Morisaki S, Smith R, Tsukada L, Sachdev S, Stevenson S, Talbot C and Zimmerman A 2023 Rapid localization and inference on compact binary coalescences with the advanced ligo-virgo-kagra gravitational-wave detector network *Phys. Rev.* D **108** 123040
[20] Morisaki S and Raymond V 2020 Rapid parameter estimation of gravitational waves from binary neutron star coalescence using focused reduced order quadrature *Phys. Rev.* D **102** 104020
[21] Talbot C, Smith R, Thrane E and Poole G B 2019 Parallelized inference for gravitational-wave astronomy *Phys. Rev.* D **100** 043030
[22] Wong K W K, Isi M and Edwards T D P 2023 Fast gravitational wave parameter estimation without compromises *Astrophys. J.* **958** 129

[23] Pathak L, Reza A and Sengupta A S 2023 Fast likelihood evaluation using meshfree approximations for reconstructing compact binary sources *Phys. Rev.* D **108** 064055
[24] Pathak L, Munishwar S, Reza A and Sengupta A S 2024 Prompt sky localization of compact binary sources using a meshfree approximation *Phys. Rev.* D **109** 024053
[25] Gabbard H, Messenger C, Siong Heng I S, Tonolini F and Murray-Smith R 2021 Bayesian parameter estimation using conditional variational autoencoders for gravitational-wave astronomy *Nat. Phys.* **18** 112–7
[26] Bhardwaj U, Alvey J, Kurt Miller B, Nissanke S and Weniger C 2023 Sequential simulation-based inference for gravitational wave signals *Phys. Rev.* D **108** 042004
[27] Dax M, Green S R, Gair J, Macke J H, Buonanno A and Schölkopf B 2021 Real-time gravitational wave science with neural posterior estimation *Phys. Rev. Lett.* **127** 241103
[28] Smith R and Thrane E 2018 Optimal search for an astrophysical gravitational-wave background *Phys. Rev.* X **8** 021019
[29] Singer L P and Price L R 2016 Rapid bayesian position reconstruction for gravitational-wave transients *Phys. Rev.* D **93** 024013
[30] Chatterjee D, Ghosh S, Brady P R, Kapadia S J, Miller A L, Nissanke S and Pannarale F 2020 A machine learning-based source property inference for compact binary mergers *Astrophys. J.* **896** 54
[31] Marx E *et al* 2024 A machine-learning pipeline for real-time detection of gravitational waves from compact binary coalescences
[32] Gabbard H, Messenger C, Siong Heng I, Tonolini F and Murray-Smith R 2020 Bayesian parameter estimation using conditional variational autoencoders for gravitational-wave astronomy
[33] Khan S, Husa S, Hannam M, Ohme F, P¥"urrer M, Jim¥'enez Forteza X and Boh¥'e A 2016 Frequency-domain gravitational waves from nonprecessing black-hole binaries. II. A phenomenological model for the advanced detector era *Phys. Rev.* D **93** 044007
[34] Buonanno A, Iyer B R, Ochsner E, Pan Y and Sathyaprakash B S 2009 Comparison of post-newtonian templates for compact binary inspiral signals in gravitational-wave detectors *Phys. Rev.* D **80** 084043
[35] LIGO Scientific Collaboration 2018 LIGO algorithm library—LALSuite Free Software (GPL)
[36] Falcon W and The PyTorch Lightning team 2024 Pytorch lightning
[37] Wu Y and He K 2018 Group normalization
[38] Kaiming H, Zhang X, Ren S and Sun J 2015 Deep residual learning for image recognition
[39] Bardes A, Ponce J and LeCun Y 2022 Vicreg: variance-invariance-covariance regularization for self-supervised learning
[40] Dax M, Green S R, Gair J, Deistler M, Schölkopf B and Macke J H 2023 Group equivariant neural posterior estimation
[41] Chatterjee D, Harris P C, Goel M, Desai M, Coughlin M W and Katsavounidis E 2023 Optimizing likelihood-free inference using self-supervised neural symmetry embeddings (arXiv:2312.07615)
[42] Papamakarios G, Nalisnick E, Jimenez Rezende D, Mohamed S and Lakshminarayanan B 2021 Normalizing flows for probabilistic modeling and inference (arXiv:1912.02762)
[43] Kingma D P, Salimans T, Jozefowicz R, Chen Xi, Sutskever I and Welling M 2017 Improving variational inference with inverse autoregressive flow (arXiv:1606.04934)
[44] Papamakarios G, Pavlakou T and Murray I 2018 Masked autoregressive flow for density estimation
[45] Germain M, Gregor K, Murray I and Larochelle H 2015 Made: masked autoencoder for distribution estimation *Proc. 32nd Int. Conf. on Machine Learning (Lille, France, 7–9 July 2015)* (*Proc. of Machine Learning Research* vol 37), ed F Bach and D Blei (PMLR) pp 881–9
[46] Bingham E, Chen J P, Jankowiak M, Obermeyer F, Pradhan N, Karaletsos T, Singh R, Szerlip P, Horsfall P and Goodman N D 2019 Pyro: deep universal probabilistic programming *J. Mach. Learn. Res.* **20** 1–6
[47] Loshchilov I and Hutter F 2019 Decoupled weight decay regularization
[48] Górski K M, Hivon E, Banday A J, Wandelt B D, Hansen F K, Reinecke M and Bartelmann M 2005 HEALPix: a framework for high-resolution discretization and fast analysis of data distributed on the sphere *Astrophys. J.* **622** 759–71
[49] Ashton G *et al* 2019 BILBY: a user-friendly Bayesian inference library for gravitational-wave astronomy *Astrophys. J. Suppl.* **241** 27
[50] Romero-Shaw I M *et al* 2020 Bayesian inference for compact binary coalescences with BILBY: validation and application to the first ligo-virgo gravitational-wave transient catalogue *Mon. Not. R. Astron. Soc.* **499** 3295–319
[51] Joshua S S 2020 dynesty: a dynamic nested sampling package for estimating Bayesian posteriors and evidences *Mon. Not. R. Astron. Soc.* **493** 3132–58
[52] Cutler C and Flanagan E É 1994 Gravitational waves from merging compact binaries: How accurately can one extract the binary's parameters from the inspiral waveform? *Phys. Rev.* D **49** 2658
[53] Nissanke S, Holz D E, Hughes S A, Dalal N and Sievers J L 2010 Exploring short gamma-ray bursts as gravitational-wave standard sirens *Astrophys. J.* **725** 496–514
[54] Saleem M *et al* 2023 Demonstration of machine learning-assisted real-time noise regression in gravitational wave detectors
[55] Raikman R *et al* 2024 GWAK: gravitational-wave anomalous knowledge with recurrent autoencoders *Mach. Learn. Sci. Tech.* **5** 025020
[56] Abbott R *et al* 2023 Open Data from the third observing run of LIGO, Virgo, KAGRA and GEO *Astrophys. J. Suppl.* **267** 29
[57] Abbott R *et al* 2021 Open data from the first and second observing runs of advanced LIGO and advanced virgo *SoftwareX* **13** 100658
[58] Lisha Li, Jamieson K, Rostamizadeh A, Gonina K, Hardt M, Recht B and Talwalkar A 2018 Massively parallel hyperparameter tuning
[59] Liaw R, Liang E, Nishihara R, Moritz P, Gonzalez J E and Stoica I 2018 Tune: a research platform for distributed model selection and training (arXiv:1807.05118)